MISSIONJOB.IN

COMPUTER INSTRUCTOR

कंप्यूटर अनुदेशक

System Analysis and Design (SAD)

- Introduction to System Analysis and Design
- Requirement Gathering and Feasibility Analysis
- Structured Analysis
- Structured Design
- Object-Oriented Modelling Using UML
- Testing
- System Implementation and Maintenance
- Other Software Development Approaches
- SDLC Phases and Models

1. Introduction to System Analysis and Design (SAD)

1.1 Basics of System

- **Definition:** System एक **interrelated components** का set है जो **common goal** प्राप्त करने के लिए एक साथ work करते हैं.
- **IPO Model:** यह एक defined structure follow करता है जिसमें Accepts input → processes → produces output की cycle होती है.
- **Examples:** Computer system (CPU, memory, I/O devices) और Banking system.

Key Characteristics of System (सिस्टम की विशेषताएँ)

- **Organization:** System के components का structured arrangement होता है.
- **Interaction:** सभी Components एक-दूसरे के साथ मिलकर work करते हैं.
- **Interdependence:** System का एक part दूसरे part पर depend करता है.
- **Integration:** सभी parts मिलकर एक unified whole (संपूर्ण इकाई) बनाते हैं.
- **Goal-oriented:** System हमेशा किसी specific objective (उद्देश्य) के लिए कार्य करता है.
- **Boundary:** यह System को उसके external environment से अलग (separate) करती है.

Components of a System

- **Input:** System में enter किया गया raw data.
- **Process:** Data को information में बदलने की processing.
- **Output:** System द्वारा produce किया गया final result.
- **Control:** यह ensure करता है कि system correctly work करे.
- **Feedback:** System performance को improve करने के लिए output का कुछ भाग वापस input में भेजा जाता है.

https://www.missionjob.in

Types of Systems: Open System vs Closed System

Feature (विशेषता)	Open System (खुली प्रणाली)	Closed System (बंद प्रणाली)
Interaction	Environment के साथ interact करता है.	Environment से कोई interaction नहीं होता.
Input/Output	External exchange (I/O) संभव है.	कोई external exchange नहीं होता.
Flexibility	यह nature में Flexible होता है.	यह पूरी तरह Fixed होता है.
Adaptability	High adaptability (बदलाव के अनुकूल).	Low adaptability.
Examples	Online shopping system, Business system.	Mechanical watch, Thermos flask.

1.2 System Analysis and Design Overview

- **System Analysis (What to do):** Existing system की study करना, problems identify करना और users से **requirement gathering** करना.
  - Example: Computerization के लिए manual library system की study करना.
- **System Design (How to do):** System के लिए technical **solution design** करना, जिसमें Database, UI और Architecture design शामिल है.
  - Example: Banking app के लिए login page और database structure create करना.
- **Objectives of SAD:** System efficiency improve करना, errors & cost reduce करना, और decision-making को support करना.
- **Importance:** यह structured software development, proper documentation और better project planning में help करता है.

https://www.missionjob.in

1.3 System Development Life Cycle (SDLC)

- **Definition:** SDLC software develop करने का एक **step-by-step process** है जो quality और structured development ensure करता है.

> 📝 **Trick to Remember SDLC Phases:**
> **P**lease **A**llow **D**evelopers **I**nto **T**he **M**eeting
> (Planning → Analysis → Design → Implementation → Testing → Maintenance).

Phases of SDLC (विस्तृत चरण):

1. **Planning Phase:** Project goals define करना, resource & cost estimation करना, और **Feasibility study** (technical, economic, operational) करना.
2. **Analysis Phase:** Current system की study करके **user requirements** collect करना और problem identification करना.
3. **Design Phase:** System structure, Architecture, UI और **Database design (schema)** तैयार करना.
4. **Implementation Phase:** Programming languages का उपयोग करके **Coding and development** start करना और hardware/software setup करना.
5. **Testing Phase:** System में errors और bugs check करना (e.g., Unit, Integration, System testing) ताकि correct working ensure हो सके.
6. **Maintenance Phase:** Deployment के बाद user support, bug fixing, updates और system improvement करना.

📝 ONE-PAGE REVISION SHEET (Compact Summary)

- **System:** Interrelated components working for a common goal using IPO (Input-Process-Output) model.
- **Characteristics:** Organization, Interaction, Interdependence, Integration, Goal-oriented, Boundary.
- **Components:** Input, Process, Output, Control, Feedback.

https://www.missionjob.in

- **Open vs Closed System:** Open system environment से interact करता है (Flexible), जबकि Closed system isolated रहता है (Fixed).
- **System Analysis:** "What system should do" (Requirement gathering & problem identification).
- **System Design:** "How system will work" (Database, UI, & Architecture design).
- **SDLC:** Step-by-step software development process.
- **SDLC Phases:**
  - Planning: Feasibility study & cost estimation.
  - Analysis: Requirement collection.
  - Design: DB & UI structure.
  - Implementation: Coding.
  - Testing: Bug finding (Unit/Integration).
  - Maintenance: Post-deployment updates.

💡 ONE-LINER REVISION (Exam Focus)

- **Q:** System performance को improve करने के लिए किस component का उपयोग होता है? **Ans: Feedback.**
- **Q:** वह System जो अपने environment के साथ interact नहीं करता, क्या कहलाता है? **Ans: Closed System** (e.g., Mechanical watch).
- **Q:** SDLC में Feasibility study (Technical, Economic) किस phase में की जाती है? **Ans: Planning Phase.**
- **Q:** Existing system की study करना और requirement gather करना किस phase का हिस्सा है? **Ans: System Analysis Phase.**
- **Q:** System का Database structure और UI design किस phase में तैयार किया जाता है? **Ans: Design Phase.**
- **Q:** Software deployment के बाद bug fixing और updates किस phase में आते हैं? **Ans: Maintenance Phase.**
- **Q:** System के components का एक-दूसरे पर निर्भर होना क्या कहलाता है? **Ans: Interdependence.**

https://www.missionjob.in

2. Requirement Gathering and Feasibility Analysis

2.1 Requirement Gathering (आवश्यकता एकत्रीकरण)

- **Definition:** Requirements मुख्य रूप से **user needs** और **system expectations** हैं, जो यह describe करती हैं कि system को क्या करना चाहिए.
- **Purpose:** यह किसी भी system development का मुख्य basis (आधार) होता है.
- **Examples:** Student system में student records store करना, Banking system में money transfer support करना.

2.1.2 Types of Requirements (आवश्यकताओं के प्रकार)

- **Functional Requirements:** यह system के **functions और operations** को define करता है, अर्थात system को क्या काम करना है (**What system does**).
  - Examples: Login system authentication, employee की salary slip generate करना.
- **Non-Functional Requirements:** यह system की **quality और performance** को define करता है, अर्थात system कैसे कार्य करेगा (**How system works**).
  - Examples: System का 2 seconds में load होना, 99% uptime availability सुनिश्चित करना.

Comparison Table: Functional vs Non-Functional

Feature (विशेषता)	Functional Requirements	Non-Functional Requirements
Focus	**What** system does (क्या करता है)	**How** system performs (कैसे कार्य करता है)
Nature	Task-based (कार्य आधारित)	Quality-based (गुणवत्ता आधारित)
Example	Login authentication, Add record	Speed, Security, 99% Uptime

https://www.missionjob.in

2.1.3 Requirement Elicitation Techniques (आवश्यकताएँ प्राप्त करने की तकनीकें)

Users से requirements collect करने के लिए उपयोग किए जाने वाले प्रमुख methods:

- **Interviews:** Users के साथ **direct discussion** करना (e.g., software needs के लिए Bank manager का interview).
- **Questionnaires:** Users को जानकारी प्राप्त करने के लिए **written set of questions** देना.
- **Observation:** Users के work और daily operations को directly **watch (देखना)** करना.
- **Document Analysis:** पुरानी प्रक्रियाओं को समझने के लिए **existing documents** की study करना.
- **JAD (Joint Application Development):** Requirements gather करने के लिए stakeholders के साथ **group discussion sessions** आयोजित करना.

2.1.4 Requirement Documentation – SRS (Software Requirement Specification)

- **Definition:** SRS एक document है जिसमें complete **system requirements** (Functional & Non-functional) लिखी होती हैं.
- **Role:** यह client और developer के बीच एक **contract (अनुबंध)** के रूप में कार्य करता है.
- **Key Contents:** Functional requirements, Non-functional requirements, System constraints, और Interface requirements.
- Examples: Online exam system या Hospital management system का SRS document.

2.2 Feasibility Study (संभाव्यता अध्ययन)

- **Definition:** System development शुरू करने से पहले यह check करना कि **project possible है या नहीं**, Feasibility Study कहलाता है.
- **Importance:** यह project management और decision making में help करता है.

https://www.missionjob.in

Types of Feasibility (संभाव्यता के प्रकार)

1. **Technical Feasibility:** Project के लिए आवश्यक **technology, hardware, software, और skills** की availability check करना (e.g., Server infrastructure उपलब्ध है या नहीं).
2. **Economic Feasibility:** Project की **Cost vs Benefit** और overall profitability evaluate करना (e.g., Software cost और expected profit की तुलना).
3. **Operational Feasibility:** System की **user acceptance, usability** और real environment में working check करना (e.g., Staff को training की आवश्यकता है या नहीं).
4. **Legal Feasibility: Legal compliance** (कानूनी अनुपालन) check करना ताकि कोई law violate ना हो (e.g., Data privacy laws और Software license usage).

2.3 Importance of Feasibility Analysis (संभाव्यता विश्लेषण का महत्व)

- **Project Viability:** यह निर्धारित करता है कि project **practical (व्यावहारिक)** है या नहीं, जिससे failed projects से बचा जा सकता है.
- **Cost-Benefit Analysis:** Development cost और expected benefits को compare करके **financial decision making** में सहायता करता है (e.g., Automation से manpower cost कम होना).
- **Risk Assessment:** Project में **possible risks (जैसे System failure या Budget overrun)** को identify करना और risk reduction planning करना.

📝 ONE-PAGE REVISION SHEET (Compact Summary)

- **Requirements:** User needs और expectations जो system development का आधार हैं.
- **Functional Req:** System के operations (What system does) - Task based (e.g., Login, add record).
- **Non-Functional Req:** System की performance (How system works) - Quality based (e.g., Speed, security).

https://www.missionjob.in

- **Elicitation Techniques:** Interviews (Direct talk), Questionnaires (Written qs), Observation (Watching), Document Analysis (Study old docs), JAD (Group discussion).
- **SRS:** Complete requirements का document जो client-developer contract के रूप में कार्य करता है.
- **Feasibility Study:** Project की possibility check करने की pre-development process.
- **Types of Feasibility:**
  - Technical: H/W, S/W, & skills check.
  - Economic: Cost vs Benefit (Profitability).
  - Operational: User acceptance & training check.
  - Legal: Laws & License compliance.

ONE-LINER REVISION (Exam Focus)

- **Q:** वह requirement जो system के specific functions (जैसे Login) को दर्शाती है, क्या कहलाती है? **Ans: Functional Requirement.**
- **Q:** System speed, security, और uptime किस प्रकार की requirement के अंतर्गत आते हैं? **Ans: Non-Functional Requirement.**
- **Q:** Requirements gather करने के लिए stakeholders के group discussion session को क्या कहा जाता है? **Ans: JAD (Joint Application Development).**
- **Q:** Client और Developer के बीच Requirement Contract के रूप में कौन सा Document कार्य करता है? **Ans: SRS (Software Requirement Specification).**
- **Q:** Development से पहले Project की Technical, Economic, और Operational viability check करना क्या कहलाता है? **Ans: Feasibility Study.**
- **Q:** Project की Cost और Expected Benefits की तुलना किस Feasibility में की जाती है? **Ans: Economic Feasibility.**
- **Q:** Software Data Privacy laws और licenses का compliance किस feasibility में check होता है? **Ans: Legal Feasibility.**

https://www.missionjob.in

3. Structured Analysis (संरचित विश्लेषण)

3.1 Introduction to Structured Analysis

- **Definition:** Structured Analysis एक **system development method** है जो system को छोटे और manageable parts में break करता है.
- **Focus:** यह मुख्य रूप से इस बात पर focus करता है कि "**what the system does**" (सिस्टम क्या करता है) और system processes को represent करने के लिए **graphical tools** का उपयोग करता है.
- **Objectives:** System requirements को clearly समझना, system का **logical model** create करना, और system design की complexity को reduce करना.
- Examples: Library system को issue, return, और search modules में break करना; Student admission process की mapping करना.

Tools Used in Structured Analysis (संरचित विश्लेषण के उपकरण)

1. **DFD** (Data Flow Diagram)
2. **Data Dictionary**
3. **Decision Trees**
4. **Decision Tables**
5. **Structured English**

3.2 Data Flow Diagram (DFD)

- **Definition:** DFD किसी system में **data flow (डेटा के मूवमेंट)** का graphical representation है.
- **Use:** यह processes, data stores, inputs, और outputs को show करता है.
- Examples: Student registration system या Banking transaction flow का diagram.

https://www.missionjob.in

Table: Symbols Used in DFD (DFD में प्रयुक्त प्रतीक)

Symbol (प्रतीक)	Name (नाम)	Meaning / Purpose (अर्थ /उद्देश्य)
→	**Data Flow**	System में **Movement of data** को दर्शाता है.
□	**Process**	Data के **Transformation** (प्रसंस्करण) को दर्शाता है.
‖	**Data Store**	System में **Storage of data** (डेटाबेस/फाइल) को दर्शाता है.
○	**External Entity**	Data का **Source या destination** (जैसे User/Customer) दर्शाता है.

Levels of DFD (DFD के स्तर)

Level	Description (विवरण)	Detail Level
Level 0 (Context Diagram)	पूरे system को **one process** के रूप में show करता है.	Very high (सबसे कम detail)
Level 1	System को **major sub-processes** में break करता है.	Medium detail
Level 2	Level 1 processes का further **decomposition** (विभाजन) करता है.	Detailed view

- Examples: Online shopping system (Level 0) → Order, payment, delivery processes (Level 1).

Advantages of DFD

- System flow को समझना easy होता है और users & developers के बीच **communication improve** करता है.
- यह clear **documentation** provide करता है और system analysis की complexity reduce करता है.

https://www.missionjob.in

3.3 Data Dictionary

- **Definition:** Data Dictionary data information का एक **central repository** है जिसमें data elements का description store होता है.
- **Components:** Data name, Data type, Size, Description, Default values, और Relationships.
  - Examples: `Student_ID = integer type, primary key` या `Name: Student_Name → varchar(50)`.
- **Importance:** यह clear data definitions provide करता है, database design में help करता है, **data inconsistency reduce** करता है, और data standardization ensure करता है.

3.4 Process Modeling Tools (प्रक्रिया मॉडलिंग उपकरण)

1. Decision Trees:

- **Definition:** Decision making के लिए एक **Tree-like structure** जो conditions और outcomes को show करता है.
- **Structure:** Root → Decision nodes → Branches → Outcomes.
- Example: Loan approval decision या Student pass/fail decision.

2. Decision Tables:

- **Definition:** Complex decision logic को solve करने के लिए **Conditions vs Actions** को दर्शाने वाली Table.
- **Structure:** इसमें Conditions column, Rules column, और Actions column होते हैं.
- Example: Insurance premium calculation या Shopping system में discount rules.

3. Structured English:

- **Definition:** Process logic को clearly represent करने के लिए **simple English + programming logic** का उपयोग.
- **Features:** इसमें IF-THEN-ELSE statements और Loop structures का use होता है, जिससे इसे समझना easy होता है.
- Example: `IF marks > 40 THEN Pass ELSE Fail` या `IF age > 18 THEN`

https://www.missionjob.in

eligible for vote.

📝 ONE-PAGE REVISION SHEET (Compact Summary)

- **Structured Analysis:** Method to break system into small parts using graphical tools (Focus on "What system does").
- **S.A. Tools:** DFD, Data Dictionary, Decision Trees, Decision Tables, Structured English.
- **DFD:** Graphical representation of data movement.
- **DFD Symbols:** → (Data Flow), □ (Process), ‖ (Data Store), ◯ (External Entity).
- **DFD Levels:** Level 0 (Context Diagram - Whole system), Level 1 (Major processes), Level 2 (Sub-processes detailed).
- **Data Dictionary:** Central repository for describing data elements (Metadata). Contains type, size, default values.
- **Decision Tree:** Tree-like conditional structure (Root → Nodes → Outcomes).
- **Decision Table:** Matrix of Conditions vs Actions for complex logic.
- **Structured English:** Logic representation using simple English and IF-THEN-ELSE.

💡 ONE-LINER REVISION (Exam Focus)

- **Q:** Structured Analysis का मुख्य focus किस पर होता है? **Ans: "What the system does"** (सिस्टम क्या करता है).
- **Q:** DFD में पूरे system को एक single process के रूप में किस level पर दर्शाया जाता है? **Ans: Level 0 (Context Diagram).**
- **Q:** DFD में Data के Transformation को दर्शाने के लिए किस symbol का उपयोग होता है? **Ans: Process (□).**
- **Q:** DFD में Data Store (Storage) को किस symbol से दर्शाया जाता है? **Ans: ‖ (Parallel lines/Data Store).**
- **Q:** Data elements की properties (Size, Type, Default values) को store करने वाला central repository क्या कहलाता है? **Ans: Data Dictionary.**

https://www.missionjob.in

4. Structured Design (संरचित डिजाइन)

4.1 Introduction to Structured Design

- **Definition:** Structured Design एक system design approach है जो analysis model को **design model** में convert करता है.
- **Focus:** यह मुख्य रूप से इस बात पर focus करता है कि "**how the system will work**" (सिस्टम कैसे कार्य करेगा).
- **Objectives:** System structure को clear बनाना, complexity को reduce करना, system efficiency improve करना, और **modular development** को support करना.
- **Principles of Good Design (अच्छे डिजाइन के सिद्धांत):** System modular होना चाहिए, प्रत्येक module की **single responsibility** होनी चाहिए, modules एक-दूसरे से **independent** होने चाहिए, और design को test व maintain करना आसान होना चाहिए.
- Examples: ATM software के लिए अलग-अलग modules (withdraw, balance, deposit) design करना; Payroll system का structure तैयार करना.

4.2 Design Approaches (डिजाइन दृष्टिकोण)

- **1. Top-Down Design:**
  - **Concept:** यह approach overall system से start होती है और main module को smaller sub-modules में break करती है (**Big → Small**).
  - Examples: Banking system को login, transaction और report modules में विभाजित करना; Library system को issue/return/search modules में तोड़ना.
- **2. Bottom-Up Design:**
  - **Concept:** यह छोटे modules से start होती है और उन्हें combine करके larger system का निर्माण करती है (**Small → Big**).
  - **Focus:** इसका मुख्य focus **reusable components** (पुनः प्रयोज्य घटकों) पर होता है.
  - Examples: पहले Login module बनाना और फिर पूरा app तैयार करना; Database module develop करके system में integrate करना.

https://www.missionjob.in

- **3. Modular Design:**
  - **Concept:** पूरे System को **independent modules** में divide किया जाता है, जहाँ प्रत्येक module एक specific function perform करता है.
  - **Advantage:** यह maintainability और reusability को improve करता है.
  - Examples: E-commerce system में payment module अलग से बनाना; ERP में report generation module बनाना.

Comparison Table: Design Approaches

Approach (दृष्टिकोण)	Description (विवरण)	Direction (दिशा)	Example (उदाहरण)
Top-Down	Big → Small	Top to bottom	ATM system design
Bottom-Up	Small → Big	Bottom to top	Reusable login module
Modular	Independent units	Any direction	E-commerce modules

4.3 Design Tools (डिजाइन के उपकरण)

- **1. Structure Charts:**
  - **Definition:** System modules की **hierarchy** (पदानुक्रम) को दर्शाने वाला diagram.
  - **Features:** यह modules के बीच relationship और **control flow** को represent करता है.
  - Examples: Banking system structure chart, Student management module hierarchy.

- **2. Flowcharts:**
  - **Definition:** किसी process steps का **graphical representation** (चित्रात्मक प्रस्तुति).
  - **Symbols:** इसमें start, process, decision, और end जैसे symbols का उपयोग होता है.

https://www.missionjob.in

- Examples: Login process flowchart, ATM withdrawal process flow.
- **3. Pseudocode:**
  - **Definition:** Program logic का **English-like representation** (यह actual programming code नहीं है).
  - **Use:** इसे किसी भी programming language में convert करना easy होता है.
  - Examples: `IF age > 18 THEN allow vote ELSE deny`, `Input marks → calculate result → display grade`.

4.4 Characteristics of Good Design (अच्छे डिज़ाइन की प्रमुख विशेषताएँ)

- **1. Low Coupling (कम कपलिंग):**
  - **Definition:** Modules के बीच **minimum dependency** (न्यूनतम निर्भरता) होनी चाहिए.
  - **Advantage:** एक module में changes करने से दूसरे modules affect नहीं होते हैं.
  - Examples: Payment module का login system से independent होना; Inventory system का billing से separate होना.
- **2. High Cohesion (उच्च संसंजन):**
  - **Definition:** प्रत्येक module को केवल एक **single task** perform करना चाहिए.
  - **Advantage:** यह system की clarity और efficiency को improve करता है.
  - Examples: Login module द्वारा केवल authentication handle करना; Report module द्वारा केवल reports generate करना.
- **3. Reusability (पुनः प्रयोज्यता):**
  - **Definition:** Modules इस प्रकार होने चाहिए कि उन्हें दूसरे systems में भी **reuse** किया जा सके.
  - **Advantage:** यह software development time को reduce करता है.
  - Examples: Login module को multiple apps में उपयोग करना; Payment gateway को विभिन्न websites में reuse करना.
- **4. Maintainability (रखरखाव योग्यता):**
  - **Definition:** System को update और fix करना आसान होना चाहिए ताकि यह **long-term usage** को support कर सके.

https://www.missionjob.in

- Examples: Single module में bug fix करना; Backend को affect किए बिना UI update करना.

📝 ONE-PAGE REVISION SHEET (Compact Summary)

- **Structured Design:** Converts analysis to design model, focuses on "How system will work".
- **Top-Down Design:** Big to Small (Main module to sub-modules).
- **Bottom-Up Design:** Small to Big (Combines small reusable modules into a system).
- **Structure Chart:** Diagram showing module hierarchy and control flow.
- **Flowchart:** Graphical representation using process and decision symbols.
- **Pseudocode:** English-like representation of logic (Not actual code).
- **Coupling:** Dependency between modules. Good design = Low Coupling (Independent modules).
- **Cohesion:** Focus on single task within a module. Good design = High Cohesion (Single responsibility).
- **Reusability & Maintainability:** Easy to reuse code in other projects and easy to fix/update.

💡 ONE-LINER REVISION (Exam Focus)

- **Q:** Analysis model को Design model में convert करने की प्रक्रिया जो "How system works" पर focus करती है, क्या कहलाती है? **Ans: Structured Design**.
- **Q:** वह Design Approach जिसमें system को Big module से Small modules की ओर break किया जाता है? **Ans: Top-Down Design**.
- **Q:** Modules की hierarchy और control flow को represent करने वाले diagram को क्या कहा जाता है? **Ans: Structure Chart**.
- **Q:** Program logic का English-like text जिसे code में बदलना आसान होता है, क्या कहलाता है? **Ans: Pseudocode**.
- **Q:** एक अच्छे Software Design में Coupling कैसी होनी चाहिए? **Ans: Low Coupling** (न्यूनतम निर्भरता).

https://www.missionjob.in

5. Object-Oriented Modelling Using UML

5.1 Introduction to UML (UML का परिचय)

- **Definition: UML (Unified Modeling Language)** एक **standard language** है जिसका उपयोग object-oriented systems को **visualize, design, और document** करने के लिए किया जाता है.
- **Purpose:** यह system requirement analysis और software design में सहायता करता है जिससे development errors reduce होती हैं.
- Examples: Banking system के लिए UML diagram बनाना; Online shopping system का design तैयार करना.
- **Importance:** यह system understanding को improve करता है, clear visual representation प्रदान करता है, और developers व users के बीच **communication** को बेहतर बनाता है (e.g., ATM system flow या Hospital system design को समझना).

UML Building Blocks (UML के निर्माण खंड)

1. **Things:** System के entities जैसे **Classes, Objects, और Components** (e.g., Class = Student, Object = Student record).
2. **Relationships:** Elements के बीच जुड़ाव जैसे **Association, Inheritance, और Dependency** (e.g., Teacher teaches Student).
3. **Diagrams:** System के structural और behavioral views को दर्शाने वाले आरेख.

5.2 Classification of UML Diagrams (UML आरेखों का प्रकार)

UML diagrams को मुख्य रूप से दो categories में बाँटा गया है:

Diagram Type (प्रकार)	Focus Area (मुख्य फोकस)	Nature (प्रकृति)	Examples (उदाहरण)
Structural Diagrams	System के **Data और Components** पर focus.	यह System का **Static structure** दर्शाता है.	Class diagram, Component diagram.

https://www.missionjob.in

Behavioral Diagrams	System के **Interaction और Flow** पर focus.	यह System का **Dynamic behavior** दर्शाता है.	Use Case (Login process), Sequence (Order flow).

5.3 Structural UML Diagrams (संरचनात्मक आरेख)

1. Class Diagram: यह system की **classes, उनके attributes, methods, और relationships** को show करता है.

- Examples: `Student class: name, roll no, marks` ; `Bank Account class: deposit(), withdraw()` .

2. Object Diagram: यह किसी specific time पर system के real objects का **snapshot** होता है, जो class diagram के **instances (उदाहरणों)** को represent करता है.

- Examples: `Student object: Rahul, Roll 101` ; `Bank account object: Account no 12345` .

3. Component Diagram: यह software modules और system components के बीच **relationships और dependencies** को दर्शाता है.

- Examples: Login module, Payment module; Database & Application components.

4. Deployment Diagram: यह hardware और software के **physical system deployment (architecture)** को show करता है.

- Examples: Server-client architecture; Cloud-based system deployment.

5.4 Behavioral UML Diagrams (व्यवहार आरेख)

1. Use Case Diagram: यह **user (actor) और system के बीच interaction** को दर्शाता है और system functionality को define करता है.

- Examples: User द्वारा ATM में login करना; Customer द्वारा online order place करना.

2. Sequence Diagram: यह objects के बीच **time order (समय क्रम)** में message flow और interactions को दर्शाता है.

https://www.missionjob.in

- Examples: ATM withdrawal के step-by-step actions; Online payment process का time sequence.

3. Activity Diagram: यह system के **workflow** को represent करता है, जो कार्यप्रणाली में **flowchart के समान** होता है.

- Examples: Order processing workflow; Student admission process.

4. State Diagram: यह किसी system object की विभिन्न **states और उसके life cycle** को show करता है.

- Examples: Order states (`pending → shipped → delivered`); Account states (`active → blocked → closed`).

5.5 Core OOP Concepts in UML (UML में OOP अवधारणाएँ)

1. Encapsulation (एनकैप्सुलेशन):

- **Definition: Data और methods को एक साथ bind करना** ताकि internal details को **hide** किया जा सके.
- Examples: Password protection system; Users से Bank account balance की internal details छिपाना.

2. Inheritance (इनहेरिटेंस):

- **Definition:** जब एक class दूसरी class की **properties inherit (प्राप्त)** करती है, जिससे **code reuse** promote होता है.
- Examples: `Vehicle` class से `Car` और `Bike`; `Employee` class से `Manager` और `Clerk` का बनना.

3. Polymorphism (पॉलीमॉर्फिज्म):

- **Definition:** Same function का **different behaviors** शो करना (इसे Method overloading/overriding भी कहते हैं).
- Examples: `Print()` function का text या image print करना; Payment method के रूप में card/cash/UPI का उपयोग.

https://www.missionjob.in

4. Abstraction (एब्स्ट्रेक्शन):

- **Definition:** केवल **essential features (आवश्यक विशेषताएँ)** को show करना और implementation (backend) details को hide करना.
- Examples: ATM interface का backend logic छिपाना; Engine की internal details जाने बिना Car drive करना.

ONE-PAGE REVISION SHEET (Compact Summary)

- **UML (Unified Modeling Language):** Object-oriented systems को visualize, design और document करने की standard language.
- **UML Blocks:** Things (Classes/Objects), Relationships (Inheritance/Dependency), Diagrams.
- **Structural Diagrams:** Static system design (e.g., Class, Object, Component, Deployment).
- **Behavioral Diagrams:** Dynamic interaction & flow (e.g., Use Case, Sequence, Activity, State).
- **Deployment Diagram:** Physical hardware/software architecture (Server-client).
- **Sequence Diagram:** Time-ordered message flow between objects.
- **Activity Diagram:** System workflow, similar to flowchart.
- **Encapsulation:** Data binding & detail hiding.
- **Inheritance:** Property inheritance for Code Reusability.
- **Polymorphism:** Same function, different behaviors (Overloading/Overriding).
- **Abstraction:** Showing essential features, hiding backend implementation logic.

https://www.missionjob.in

💡 ONE-LINER REVISION (Exam Focus)

- **Q:** Object-oriented systems को visualize और document करने के लिए किस Standard Language का उपयोग किया जाता है? **Ans: UML (Unified Modeling Language)**.
- **Q:** System की static structure (Data/Components) को दर्शाने वाले diagrams क्या कहलाते हैं? **Ans: Structural Diagrams** (e.g., Class Diagram).
- **Q:** किसी specific time पर system के real objects का snapshot कौन सा diagram दिखाता है? **Ans: Object Diagram**.
- **Q:** Hardware और Software के physical system architecture (Deployment) को कौन दर्शाता है? **Ans: Deployment Diagram**.
- **Q:** Actor (User) और System के बीच के interaction को किस diagram द्वारा दर्शाया जाता है? **Ans: Use Case Diagram**.
- **Q:** Objects के बीच Time-ordered sequence में message flow कौन सा diagram दिखाता है? **Ans: Sequence Diagram**.
- **Q:** System workflow को flowchart के रूप में represent करने वाला diagram कौन सा है? **Ans: Activity Diagram**.
- **Q:** Data और Methods को एक साथ bind करके internal details hide करने की OOP concept क्या है? **Ans: Encapsulation**.
- **Q:** Code reuse को promote करने के लिए एक Class द्वारा दूसरी Class की properties प्राप्त करना क्या कहलाता है? **Ans: Inheritance**.
- **Q:** Backend implementation logic को छिपाकर केवल essential features show करना क्या कहलाता है? **Ans: Abstraction**.

https://www.missionjob.in

6. Software Testing (सॉफ्टवेयर टेस्टिंग)

6.1 Introduction to Software Testing

- **Definition:** Software Testing एक process है जिसमें software में errors और defects check किए जाते हैं ताकि यह सुनिश्चित हो सके कि system requirements के अनुसार काम कर रहा है.
- **Objectives:** Bugs find करना, software quality ensure करना, और reliability व performance improve करना.
- **Importance:** यह maintenance cost reduce करता है, user satisfaction बढ़ाता है, और system failure को prevent करता है.
- Examples: Login functionality check करना; Payment failure cases की testing करना.

6.2 Types of Testing (टेस्टिंग के प्रकार)

- **1. Unit Testing:** यह individual module या component की testing है, जो developers द्वारा की जाती है.
- **2. Integration Testing:** यह combined modules की testing है, जो modules के बीच data flow check करती है.
- **3. System Testing:** Complete system की as a whole (संपूर्ण रूप में) testing, जो सुनिश्चित करती है कि system requirements को पूरा करता है.
- **4. Acceptance Testing:** End users या client द्वारा की जाने वाली testing जो confirm करती है कि system उपयोग के लिए तैयार है.

Comparison Table: Types of Testing

Type (प्रकार)	Level (स्तर)	Done By (किसके द्वारा)	Focus (मुख्य बिंदु)
Unit Testing	Small module	Developer	Individual function
Integration Testing	Combined modules	Tester	Module interaction

https://www.missionjob.in

System Testing	Full system	QA team	Overall system
Acceptance Testing	Final stage	User/Client	Requirement satisfaction

6.3 Testing Techniques (टेस्टिंग तकनीकें)

- **1. Black Box Testing:** Internal code knowledge के बिना testing; मुख्य रूप से input-output behavior पर focus करती है.
- **2. White Box Testing:** Internal code structure और logic की testing; Developers द्वारा paths और loops check करने के लिए की जाती है.
- **3. Grey Box Testing:** Black box + White box का combination; Partial knowledge (आंशिक जानकारी) के साथ internal system testing.

Comparison Table: Testing Techniques

Technique (तकनीक)	Knowledge (जानकारी)	Focus (फोकस)	Tester (परीक्षक)
Black Box	No code knowledge	Input-output	Tester
White Box	Full code knowledge	Logic	Developer
Grey Box	Partial knowledge	Both	Tester/Developer

6.4 Testing Levels (टेस्टिंग के स्तर)

- **1. Alpha Testing:** Developer's site पर release से पहले होने वाली internal testing. Example: Software inside company testing.
- **2. Beta Testing:** User's environment में real users द्वारा testing, जहाँ final feedback collect किया जाता है. Example: WhatsApp beta version testing.

6.5 Debugging (डिबगिंग)

- **Definition:** Testing के बाद detected errors (bugs) को find और fix करने की process Debugging कहलाती है.

https://www.missionjob.in

- **Types of Errors (त्रुटियों के प्रकार):**
  - **Syntax Errors:** Code में Grammar mistakes (e.g., Missing semicolon).
  - **Logical Errors:** Wrong logic के कारण गलत output आना (e.g., Wrong formula in marks calculation).
  - **Runtime Errors:** Program execution के दौरान आने वाली errors (e.g., Division by zero).
- **Debugging Process:** Error Identify करना → Locate करना → Analyze करना → Bug Fix करना → Retest करना.

ONE-PAGE REVISION SHEET (Compact Summary)

- **Software Testing:** Process to check errors and ensure software quality.
- **Unit Testing:** Individual module test by Developer.
- **Integration Testing:** Module interaction & data flow test.
- **System Testing:** Complete overall system test by QA team.
- **Acceptance Testing:** Final user/client approval test.
- **Black Box Testing:** Tests input-output without code knowledge.
- **White Box Testing:** Tests internal code logic by Developer.
- **Grey Box Testing:** Partial code knowledge testing.
- **Alpha Testing:** Internal testing at developer's site.
- **Beta Testing:** Real user testing in real environment for feedback.
- **Debugging:** Process of finding and fixing bugs.
- **Errors:** Syntax (grammar rule), Logical (wrong output), Runtime (crash during execution).

https://www.missionjob.in

💡 ONE-LINER REVISION (Exam Focus)

- **Q:** Testing की वह process जो individual module या function पर focus करती है, क्या कहलाती है? **Ans: Unit Testing.**
- **Q:** Modules को combine करने के बाद उनके बीच data flow check करना किस testing में आता है? **Ans: Integration Testing.**
- **Q:** Client द्वारा system की final checking (readiness confirmation) को क्या कहते हैं? **Ans: Acceptance Testing.**
- **Q:** Internal code structure की जानकारी के बिना input-output behavior test करना क्या कहलाता है? **Ans: Black Box Testing.**
- **Q:** Developer द्वारा internal code paths और logic को test करना किस तकनीक का हिस्सा है? **Ans: White Box Testing.**
- **Q:** Release से पहले Developer site पर की जाने वाली internal testing को क्या कहते हैं? **Ans: Alpha Testing.**
- **Q:** Real users द्वारा उनके environment में की जाने वाली testing (जैसे WhatsApp Beta) क्या कहलाती है? **Ans: Beta Testing.**
- **Q:** Errors (Bugs) को find करके fix करने की प्रक्रिया को क्या कहा जाता है? **Ans: Debugging.**
- **Q:** Code में missing semicolon किस प्रकार का error है? **Ans: Syntax Error.**

https://www.missionjob.in

7. System Implementation and Maintenance

7.1 System Implementation (सिस्टम कार्यान्वयन)

- **Definition:** System Implementation वह process है जिसमें नए system को **install और deploy** किया जाता है, अर्थात् Design को **working system** में convert किया जाता है.
- **Key Activities:** इसमें **installation, training, data conversion, और testing** शामिल हैं.
- Examples: College में online examination system deploy करना या Bank में banking software install करना.

7.1.2 Methods of Implementation (कार्यान्वयन की विधियाँ)

- **1. Direct Conversion (सीधा रूपांतरण):**
  - **Concept:** Old system को तुरंत हटाकर **completely एक साथ नया system** शुरू कर दिया जाता है.
  - **Advantages/Disadvantages:** यह Fast और Low cost है, लेकिन इसमें **High risk** होता है क्योंकि कोई backup system नहीं होता.
  - Example: Manual billing से तुरंत software पर switch करना.
- **2. Parallel Conversion (समानांतर रूपांतरण):**
  - **Concept:** Old और New systems दोनों को **एक साथ (together) run** किया जाता है और output compare किया जाता है.
  - **Advantages/Disadvantages:** यह Very safe है (error checking possible), लेकिन बहुत **Costly और Slow** होता है.
  - Example: Manual और computerized payroll system को एक साथ चलाना.
- **3. Phased Conversion (चरणबद्ध रूपांतरण):**
  - **Concept:** System को एक साथ लागू न करके **step-by-step (phases/modules)** में implement किया जाता है.
  - **Advantages/Disadvantages:** Low risk और easy control, लेकिन **Slow process**.
  - Example: Library system के modules को एक-एक करके लागू करना.

https://www.missionjob.in

- **4. Pilot Conversion (पायलट रूपांतरण):**
  - **Concept:** New system को पहले केवल **एक small unit (जैसे एक branch)** में introduce किया जाता है. Successful होने पर हर जगह implement होता है.
  - **Advantages/Disadvantages:** Risk reduce होता है (Real testing environment), लेकिन Time-consuming है.
  - Example: Bank की एक branch में नया system test करना.

Comparison Table: Implementation Methods (कार्यान्वयन विधियों की तुलना)

Method (विधि)	Risk (जोखिम)	Speed (गति)	Cost (लागत)	Description (विवरण)
Direct	**High**	Fast	Low	Immediate switch (तुरंत बदलाव).
Parallel	**Low**	Slow	**High**	Both systems run (दोनों एक साथ).
Phased	Medium	Slow	Medium	Step-by-step (चरणबद्ध).
Pilot	Low	Medium	Medium	One unit first (पहले एक यूनिट में).

7.2 System Maintenance (सिस्टम रखरखाव)

- **Definition:** System Maintenance वह process है जिसमें implementation के बाद system को **update और improve** किया जाता है, ताकि errors fix हों और performance बेहतर हो.
- Examples: Software errors (bugs) fix करना या Mobile App में new features add करना.

Types of Maintenance (रखरखाव के प्रकार)

- **1. Corrective Maintenance (सुधारात्मक रखरखाव):**
  - **Concept:** Problem detect होने के बाद code में मौजूद **errors और bugs को fix करना**.
  - Examples: Login failure या payroll calculation error को सही करना.

https://www.missionjob.in

- **2. Adaptive Maintenance (अनुकूली रखरखाव):**
  - **Concept:** System को नए environment (Hardware/Software changes) के अनुसार **modify और adjust** करना.
  - Examples: नए OS या नए Browser compatibility के लिए software update करना.
- **3. Perfective Maintenance (पूर्णतावादी रखरखाव):**
  - **Concept:** System performance, user experience और **features को improve** करना.
  - Examples: App में नए features (enhancements) जोड़ना या System speed बढ़ाना.
- **4. Preventive Maintenance (निवारक रखरखाव):**
  - **Concept: Future problems से बचने** के लिए system reliability बढ़ाना.
  - Examples: Code optimization करना और regular system updates देना.

Comparison Table: Maintenance Types (रखरखाव प्रकार की तुलना)

Type (प्रकार)	Purpose (उद्देश्य)	Example (उदाहरण)
Corrective	Fix errors	Bug fixing
Adaptive	Environment changes	OS update
Perfective	Improvement	New features
Preventive	Avoid future issues	Optimization

7.3 Post-Implementation Review (कार्यान्वयन के बाद की समीक्षा)

- **1. System Evaluation (सिस्टम मूल्यांकन):** यह check करना कि system ने अपनी **requirements को meet** किया है या नहीं (e.g., Banking system accuracy check).
- **2. User Feedback (उपयोगकर्ता प्रतिक्रिया):** System improvement के लिए users से **opinions और suggestions** collect करना (e.g., App usability feedback).
- **3. Performance Analysis (प्रदर्शन विश्लेषण):** System की **speed, efficiency, और reliability** को measure करके system weaknesses को identify करना (e.g., Server response time monitor करना).

https://www.missionjob.in

ONE-PAGE REVISION SHEET (Compact Summary)

- **System Implementation:** Design को Working System में convert करने (install/deploy) की process.
- **Direct Conversion:** Immediate replace old system. High Risk, Fast.
- **Parallel Conversion:** Run both Old + New system. Low Risk, High Cost.
- **Phased Conversion:** Step-by-step module implement. Medium Risk, Slow.
- **Pilot Conversion:** Test in one small unit first. Low Risk, Real test.
- **System Maintenance:** Post-implementation updating and improving process.
- **Corrective Maintenance:** Fixing existing Errors/Bugs.
- **Adaptive Maintenance:** Adapting to New Environment/OS.
- **Perfective Maintenance:** Adding New Features / Performance Improvement.
- **Preventive Maintenance:** Code Optimization to Avoid Future Issues.
- **Post-Implementation Review:** System Evaluation, User Feedback & Performance Analysis (Speed/Load test).

ONE-LINER REVISION (Exam Focus)

- **Q:** Old system को तुरंत बंद करके New System को एकदम से लागू करना क्या कहलाता है? **Ans: Direct Conversion** (High Risk).
- **Q:** वह Conversion Method कौन सा है जो सबसे Costly है क्योंकि उसमें Old और New दोनों systems एक साथ चलते हैं? **Ans: Parallel Conversion**.
- **Q:** New system को पूरी संस्था में लागू करने से पहले किसी एक Branch या Department में लागू करके check करना क्या कहलाता है? **Ans: Pilot Conversion**.
- **Q:** System implementation के बाद Software में आने वाले Bugs और Errors को fix करना किस maintenance में आता है? **Ans: Corrective Maintenance**.
- **Q:** System को नए Operating System (OS) या Hardware के अनुसार बदलना क्या कहलाता है? **Ans: Adaptive Maintenance**.
- **Q:** System में नए Features add करना और उसकी overall quality improve करना किस प्रकार का Maintenance है? **Ans: Perfective Maintenance**.
- **Q:** भविष्य में आने वाली समस्याओं को रोकने के लिए Code Optimize करना क्या कहलाता है? **Ans: Preventive Maintenance**.

https://www.missionjob.in

8. Other Software Development Approaches

8.1 Software Development Models (सॉफ्टवेयर विकास मॉडल)

- **1. Waterfall Model:**
  - **Concept:** यह एक **linear and sequential model** है जिसमें एक phase पूरी तरह से complete होने के बाद ही अगला phase start होता है.
  - **Phases Flow:** Requirement → Design → Implementation → Testing → Maintenance.
  - **Features:** यह simple है, phases के बीच कोई overlap नहीं होता, और **small projects** (जैसे Payroll system) के लिए suitable है.
- **2. Incremental Model:**
  - **Concept:** System को **small increments (छोटे हिस्सों)** में develop किया जाता है.
  - **Features:** प्रत्येक increment system में नई functionality add करता है जिससे partial system की **early delivery** संभव होती है, और changes manage करना आसान होता है.
  - Example: Banking app में step-by-step features add करना.
- **3. Spiral Model:**
  - **Concept:** यह **iterative development और risk analysis** का बेहतरीन combination है.
  - **Phases:** Planning, Risk Analysis, Engineering, और Evaluation.
  - **Features:** पूरी Process **spirals (cycles)** में repeat होती है और यह **large & complex projects** (जैसे Defense systems) के लिए सबसे suitable है.
- **4. V-Model:**
  - **Concept:** यह Waterfall model का extension है जहाँ Development और Testing **parallel (समानांतर)** रूप से plan की जाती हैं.
  - **Structure:** इसमें Verification (Left side) और Validation (Right side) का V-shape structure बनता है.
  - **Advantage:** Strict testing के कारण errors का **early detection** होता है.

https://www.missionjob.in

8.2 Agile Methodology (एजाइल मेथोडोलॉजी)

- **8.2.1 Agile Principles (सिद्धांत):** यह **flexibility** और **continuous delivery** पर focus करता है, customer satisfaction बढ़ाता है, और development के दौरान **changing requirements** का welcome करता है.
- **8.2.2 Scrum Framework:**
  - **Concept:** यह **sprints (short cycles)** पर आधारित Agile method है.
  - **Roles:** इसमें Product Owner, Scrum Master, और Development Team प्रमुख भूमिका निभाते हैं.
  - **Features:** इसमें Daily meetings (stand-up) और sprint planning होती है.
- **8.2.3 Kanban Model:**
  - **Concept:** यह एक **visual workflow management system** है जिसमें कार्य को Boards (To Do, In Progress, Done) पर दर्शाया जाता है.
  - **Features:** यह **work in progress (WIP) को limit** करता है और continuous delivery सुनिश्चित करता है.

8.3 Rapid Application Development (RAD)

- **Concept:** यह prototyping और user feedback पर आधारित **fast development model** है.
- **Features:** इसमें planning कम होती है और development पर ज्यादा focus किया जाता है.
- **Advantages:** Faster development, high user involvement, reduced development time, और यह changes के लिए पूरी तरह flexible है.

8.4 Prototyping Model (प्रोटोटाइपिंग मॉडल)

- **1. Throwaway Prototyping:** Final system बनने के बाद prototype को **discard (नष्ट)** कर दिया जाता है.
- **2. Evolutionary Prototyping:** Prototype को लगातार **improve** करके final system में convert किया जाता है.
- **Advantages:** Users को requirement की better understanding मिलती है और risk reduce होता है.

https://www.missionjob.in

- **Disadvantages:** यह **time-consuming** हो सकता है, cost बढ़ सकती है, और इसमें poor documentation का risk रहता है.

8.5 Comparison of Models (मॉडल्स की तुलना)

Table 1: Waterfall vs Agile

Feature (विशेषता)	Waterfall Model	Agile Methodology
Approach	Linear (सीधा)	Iterative (चक्रीय)
Flexibility	Low (कम)	High (अधिक)
Customer Involvement	Low (शुरुआत में ही)	High (लगातार)
Delivery	Final Product (अंत में)	Continuous (निरंतर)

Table 2: Spiral vs Incremental

Feature (विशेषता)	Spiral Model	Incremental Model
Main Focus	Risk analysis	Feature delivery
Complexity	High	Medium
Best Usage	Large projects	Medium projects

Table 3: RAD vs Prototyping

Feature (विशेषता)	RAD (Rapid App Dev)	Prototyping
Speed	Very fast (बहुत तेज)	Moderate (मध्यम)
Primary Focus	Rapid development	Requirement clarity
Best Usage	Business apps	UI & requirement understanding

https://www.missionjob.in

📝 ONE-PAGE REVISION SHEET (Compact Summary)

- **Waterfall:** Linear, sequential, no overlap, best for small projects.
- **Incremental:** Development in small parts (increments) for early delivery.
- **Spiral:** Iterative + Risk Analysis (Planning, Risk, Engineering, Evaluation), best for large projects.
- **V-Model:** Verification and Validation, parallel testing with development.
- **Agile:** Highly flexible, iterative, welcomes changing requirements, continuous delivery.
- **Scrum:** Agile framework working in short cycles called Sprints, involves Scrum Master & Product Owner.
- **Kanban:** Visual board system (To Do, In Progress, Done) for workflow, limits WIP.
- **RAD:** Rapid Application Development, fast-paced, prototype-based, high user feedback.
- **Throwaway Prototype:** Discarded after use.
- **Evolutionary Prototype:** Improved into the final product.

https://www.missionjob.in

💡 ONE-LINER REVISION (Exam Focus)

- **Q:** वह कौन सा Model है जिसमें एक phase के complete होने के बाद ही अगला phase start होता है? **Ans: Waterfall Model.**
- **Q:** High Risk वाले Large & Complex Projects के लिए सबसे suitable model कौन सा है? **Ans: Spiral Model.**
- **Q:** Development (Verification) और Testing (Validation) का समानांतर (parallel) execution किस model में होता है? **Ans: V-Model.**
- **Q:** Agile methodology में use होने वाले short development cycles को क्या कहा जाता है? **Ans: Sprints.**
- **Q:** Visual boards का उपयोग करके Work in Progress (WIP) को limit करने वाला Agile framework कौन सा है? **Ans: Kanban Model.**
- **Q:** Fast development और User Feedback पर आधारित approach जिसमें planning कम होती है, क्या कहलाती है? **Ans: RAD (Rapid Application Development).**
- **Q:** वह prototype जिसे requirement समझने के बाद final system बनने पर नष्ट कर दिया जाता है? **Ans: Throwaway Prototyping.**
- **Q:** Customer की changing requirements को easily accept करने वाली methodology कौन सी है? **Ans: Agile Methodology.**

https://www.missionjob.in

9. Final Mega Revision & Quick Recap

9.1 SDLC and Requirement Analysis (SDLC और आवश्यकताएं)

- **SDLC Definition:** Software develop करने का एक **step-by-step process**, जिसे System Development Life Cycle कहा जाता है.
- **SDLC Phases:** Planning (Feasibility study) → Analysis (Requirement gathering) → Design (System design) → Implementation (Coding) → Testing (Error checking) → Maintenance (Updates & fixes).
  - Examples: Banking software या College management system develop करना.
- **Requirements Types:**
  - **Functional:** System क्या करता है (**What system does**), जैसे Login system या Report generation.
  - **Non-Functional:** System की quality कैसी है (**System quality**), जैसे Fast response time और High security.
- **Feasibility Study (संभाव्यता अध्ययन):**
  - Technical: Technology availability (e.g., Server check).
  - Economic: Cost vs benefit (e.g., Budget approval).
  - Operational: User acceptance check करना.

9.2 System Modeling & Design (सिस्टम मॉडलिंग और डिजाइन)

- **DFD (Data Flow Diagram):** System में **data flow (डेटा प्रवाह)** को show करने वाला diagram.
  - Symbols: Process (Rectangle), Data Flow (Arrow), Data Store (Parallel lines), External Entity (Square).
  - Levels: Level 0 (Whole system), Level 1 (Main processes), Level 2 (Detailed processes).
- **Data Dictionary:** Data information का **Central store** जिसमें Data name, type, size और description store होता है (e.g., `Student_ID: integer`).

https://www.missionjob.in

- **Structured Design Principles:**
  - **Cohesion:** Module की **internal strength**; एक module को केवल एक task (one task only) करना चाहिए. (High Cohesion = Good).
  - **Coupling:** Modules के बीच **Dependency**; इसे low होना चाहिए (minimum dependency).
- **UML Diagrams (Unified Modeling Language):**
  - Use Case Diagram: User और system के बीच **interaction** (e.g., ATM use case).
  - Class Diagram: Classes, उनके attributes और **relationships** दर्शाता है.
  - Sequence Diagram: Objects के बीच **Time-based interaction** (e.g., Online order sequence).

9.3 Software Testing & Debugging (सॉफ्टवेयर टेस्टिंग और डिबगिंग)

Table: Black Box vs White Box Testing

Feature (विशेषता)	Black Box Testing	White Box Testing
Knowledge	No code knowledge (कोड की जानकारी नहीं).	Code knowledge required.
Focus	Input-output behavior.	Internal logic & loops.
Tester	Done by Tester (टेस्टर द्वारा).	Done by Developer.
Examples	Login testing.	Loop testing.

- **Testing Levels:**
  - Unit Testing: Single module testing.
  - Integration Testing: Combined modules testing.
  - System Testing: Full overall system testing (e.g., Full banking system test).
  - Acceptance Testing: Final User/client testing.

https://www.missionjob.in

- **Debugging:** Errors को **find और fix** करने का process.
- **Error Types: Syntax** (Grammar error, e.g., missing semicolon), **Logical** (Wrong logic/formula), **Runtime** (Execution error).

9.4 Implementation, Maintenance & Models (लागू करना, रखरखाव और मॉडल)

- **Implementation Methods:**
  - Direct Conversion: Old system से immediate switch.
  - Parallel: Old + new system together.
  - Phased: Step-by-step conversion.
  - Pilot: One unit first (e.g., Bank branch migration).
- **Maintenance Types:**
  - Corrective: Bug fixing.
  - Adaptive: Environment changes.
  - Perfective: Feature improvement.
  - Preventive: Future issue prevention.

Table: Waterfall vs Agile Methodology

Feature (विशेषता)	Waterfall Model	Agile Model
Approach	Linear (सीधा)	Iterative + flexible (चक्रीय)
Flexibility	Low (कम)	High (अधिक)
Delivery	Final (अंत में)	Continuous (निरंतर)

- **Other Development Models:**
  - Spiral Model: Risk-based iterative model (e.g., Defense system).
  - V-Model: Testing parallel with development (e.g., Banking system).
  - Incremental: System developed in parts.
  - RAD: Fast development model.
  - Prototyping: Final system से पहले Trial version (Types: Throwaway, Evolutionary).

https://www.missionjob.in

- **Post-Implementation Review:** Deployment के बाद System evaluation, Performance analysis, और User feedback (e.g., checking system speed).

📝 ONE-PAGE REVISION SHEET (Compact Summary)

- **SDLC:** Software का complete development cycle.
- **DFD:** System में data flow visualization.
- **UML:** System design के लिए system modeling language.
- **Testing:** Errors और bugs को detect करने का process.
- **Maintenance:** Deployment के बाद system improvement और updates.
- **Design Rule:** High Cohesion (One task per module) और Low Coupling (Independent modules) Best होते हैं.
- **RAD & Prototype:** Fast development के लिए RAD, और requirements समझने के लिए Prototyping (Trial version) का उपयोग.

💡 ONE-LINER REVISION (Exam Focus)

- **Q:** Module की internal strength और single task capability क्या कहलाती है? **Ans: Cohesion**.
- **Q:** Modules के बीच की Dependency को क्या कहा जाता है? **Ans: Coupling**.
- **Q:** Data details (Name, type, size) स्टोर करने वाला central store क्या है? **Ans: Data Dictionary**.
- **Q:** DFD में Process को दर्शाने के लिए किस shape का प्रयोग होता है? **Ans: Rectangle**.
- **Q:** Internal Code knowledge के बिना Input-Output check करने वाली testing कौन सी है? **Ans: Black Box Testing**.
- **Q:** Development के parallel testing किस model में की जाती है? **Ans: V-Model**.
- **Q:** High risks वाले projects के लिए कौन सा Model best है? **Ans: Spiral Model (Risk-based)**.
- **Q:** Deployment के बाद system errors (bugs) को fix करना किस maintenance में आता है? **Ans: Corrective Maintenance**.

https://www.missionjob.in